Feb. 2010 CHINESE JOURNAL OF ENGINEERING MATHEMATICS Vol. 27 No. 1

Article ID:1005-3085(2010)01-0152-09

Rescheduling to Minimize Total Weighted Completion Time under Anticompatible Job Systems*

MU Yun-dong[1,2], GU Cun-chang[1,2], ZHOU Wei[1], CHENG Yao[1]

(1- College of Science, Henan University of Technology, Zhengzhou 450001;

2- Department of Mathematics, Zhengzhou University, Zhengzhou 450052)

Abstract: For the rescheduling problem on a single machine, a set of original jobs has already been scheduled to minimize some cost, then a new set of jobs arrive and create a disruption. The rescheduling problem in this paper is to minimize total weighted completion time under a limit of the maximum disruption when p_j and w_j of the jobs are anticompatible. In this paper, we consider the above rescheduling problem for jobs on a single machine. For the rescheduling problems under a limit of the maximum sequence disruption, or the maximum time disruption, or the total sequence disruption and the total time disruption, depending on their structure properties, we apply the dynamic programming method and give a polynomial time or pseudopolynomial time algorithm, respectively.

Keywords: rescheduling; single machine; completion time; disruption; anticompatible

Classification: AMS(2000) 90B35; 90C27 **CLC number:** O223 **Document code:** A

1 Introduction and problem formulation

For the rescheduling problem on a single machine, a set of original jobs has already been scheduled to minimize some cost, then a new set of jobs arrive and create a disruption. The decision maker needs to insert the new jobs into the existing schedule without excessively disrupting it.

There are several references about rescheduling approaches. Wu et al.[1] consider the problem of rescheduling a job shop after a disruption to minimize the makespan and the disruption from the original schedule. Unal et al.[2] consider a single machine with newly arrived jobs that have setup times depending on their part types. They considered inserting new jobs into the original schedule so as to minimize the total weighted completion time or makespan of the new jobs without incurring additional setups or causing jobs to be late. Li[3] consider the problem of parallel batch jobs with three hierarchical criteria on a single machine. Mu and Gu[4] consider the rescheduling problem with the some fixed ordered jobs. Hall and Potts[5] consider the problem of rescheduling of a single machine with newly arrived jobs to minimize the maximum lateness and the total completion time under a limit of the disruption from the

Received: 08 Apr 2008. **Biography**: Mu Yundong (Born in 1965), Male, Doctor, Professor. Research
Accepted: 24 Apr 2009. field: combinatorial optimization.

***Foundation item:** NSFC (10671183); NSFHN (082300410190); NSF of the Education Department of Henan Province (2008A110004); the Science Foundation (07XJC002) and Doctor Science Foundation of Henan University of Technology.

original scheduling. Yuan[6] and Mu[7] consider the problem of rescheduling of a single machine with release dates to minimize makespan under a limit of the maximum sequence disruption and rescheduling jobs to minimize maximum lateness and total tardiness under compatible job systems.

By Hall and Potts[5], the rescheduling problem for jobs on a single machine can be stated as follows.

Let $\mathcal{J}_O = \{J_1, \cdots, J_{n_O}\}$ denote a set of original jobs processed nonpreemptively on a single machine. In this model, we assume that the jobs in $\mathcal{J}_O$ have been scheduled optimally to minimize some classical objective and that π^* is an optimal schedule. Let $\mathcal{J}_N = \{J_{n_O+1}, \cdots, J_{n_O+n_N}\}$ denotes a set of new jobs. Denote $\mathcal{J} = \mathcal{J}_O \cup \mathcal{J}_N$. Each job $J_j \in \mathcal{J}$ has an integral processing time $p_j \geq 0$ and an integer weight $w_j \geq 0$. We assume that the new jobs' information (processing times and release dates) becomes known at time zero after a schedule for the jobs of $\mathcal{J}_O$ being determined, but before processing begins. Let

$$n = n_O + n_N, \quad P_O = \sum_{J_j \in \mathcal{J}_O} p_j \quad \text{and} \quad P_N = \sum_{J_j \in \mathcal{J}_N} p_j.$$

Let π^* and σ^* denote an optimal schedule of the jobs of $\mathcal{J}_O$ and $\mathcal{J}$, respectively. For any schedule σ of the jobs in $\mathcal{J}$, we define the following variables.

- $S_j(\sigma)$ is the starting processing time of job $J_j \in \mathcal{J}$.
- $C_j(\sigma) = S_j(\sigma) + p_j$ is the completing time of job $J_j \in \mathcal{J}$.
- $D_j(\pi^*, \sigma)$ is the sequence disruption of job $J_j \in \mathcal{J}_O$, i.e., if J_j is the x-th job in π^* and the y-th job in σ, respectively, then $D_j(\pi^*, \sigma) = |y - x|$.
- $\Delta_j(\pi^*, \sigma) = |C_j(\sigma) - C_j(\pi^*)|$ is the time disruption of job $J_j \in \mathcal{J}_O$.

When there is no ambiguity, the above three parameters are simplified to C_j, $D_j(\pi^*)$, and $\Delta_j(\pi^*)$, respectively.

The standard classification scheme for scheduling problems[8,9] is a three-field classification $\alpha|\beta|\gamma$, where α indicates the scheduling environment, β describes the job characteristics or restrictive requirements, and γ defines the optimality criterion. Here we consider only the single machine problems, thus implying that $\alpha = 1$. Under β, we indicate a constraint on the amount of disruption which is applicable. Such constraints include the following four forms.

- $D_{\max}(\pi^*) \leq k : \max_{J_j \in \mathcal{J}_O}\{D_j(\pi^*)\} \leq k$, the maximum sequence disruption of the jobs cannot exceed k.
- $\sum D_j(\pi^*) \leq k : \sum_{J_j \in \mathcal{J}_O} D_j(\pi^*) \leq k$, the total sequence disruption of the jobs cannot exceed k.
- $\Delta_{\max}(\pi^*) \leq k : \max_{J_j \in \mathcal{J}_O}\{\Delta_j(\pi^*)\} \leq k$, the maximum time disruption of the jobs cannot exceed k.
- $\sum \Delta_j(\pi^*) \leq k : \sum_{J_j \in \mathcal{J}_O} \Delta_j(\pi^*) \leq k$, the total time disruption of the jobs cannot exceed k.

For the rescheduling problem for jobs on a single machine to minimize total weighted completion time under a limit on the disruption constraints, we have the following four variations

$$1 \,|\, D_{\max}(\pi^*) \leq k \,|\, \sum w_j C_j, \quad 1 \,\Big|\, \sum D_j(\pi^*) \leq k \,\Big|\, \sum w_j C_j,$$

$$1 \,|\, \Delta_{\max}(\pi^*) \leq k \,|\, \sum w_j C_j, \quad 1 \,\Big|\, \sum \Delta_j(\pi^*) \leq k \,\Big|\, \sum w_j C_j.$$

In this paper, we show that the scheduling problems

$$1 \mid D_{\max}(\pi^*) \le k \mid \sum w_j C_j, \quad 1 \mid \Delta_{\max}(\pi^*) \le k \mid \sum w_j C_j,$$

$$1 \Big| \sum D_j(\pi^*) \le k \Big| \sum w_j C_j, \quad 1 \Big| \sum \Delta_j(\pi^*) \le k \Big| \sum w_j C_j$$

can be solved either in polynomial time or in pseudopolynomial time, when p_j and w_j of the jobs are **anticompatible**, i.e., $p_i \le p_j$ and $w_i \ge w_j$. Under β, we indicate a constraint on the job system, denoted by anti-(p_j, w_j).

2 Preliminaries

The following classical scheduling rule will be used.

SWPT Rule In the smallest ratio of p_j/w_j (SWPT) rule, jobs are sequenced in nondecreasing order of the ratio of their processing time to weight.

Smith shows that the SWPT rule provides an optimal schedule for the classical scheduling problem $1\|\sum w_j C_j$. Thus, we assume that the jobs of $\mathcal{J}_O$ are indexed and sequenced in the SWPT order in π^*, i.e.,

$$p_1/w_1 \le p_2/w_2 \le \cdots \le p_{n_O}/w_{n_O}.$$

For simplicity, we also assume that the new jobs are indexed in SWPT order, i.e.,

$$p_{n_O+1}/w_{n_O+1} \le p_{n_O+2}/w_{n_O+2} \le \cdots \le p_{n_O+n_N}/w_{n_O+n_N}.$$

In the following we will show that the SWPT rule applies to several of the rescheduling problems, and prove several basic results that are useful for designing algorithms.

Lemma 1 Consider two problems

$$1 \mid anti-(p_j, w_j), D_{\max}(\pi^*) \le k \mid \sum w_j C_j, \quad 1 \mid anti-(p_j, w_j), \Delta_{\max}(\pi^*) \le k \mid \sum w_j C_j,$$

where there is an optimal priority index sequencing rule for the jobs of $\mathcal{J}_O$ that gives the same ordering for the jobs of $\mathcal{J}_O$ in π^* and in σ^*. Then, each of the above two problems has an optimal schedule with no idle time between jobs, and

(a) a schedule for problem $1|D_{\max}(\pi^*) \le k|\sum w_j C_j$ is feasible if and only if the number of jobs of $\mathcal{J}_N$ scheduled before the last job of $\mathcal{J}_O$ is less than or equal to k;

(b) a schedule for problem $1|\Delta_{\max}(\pi^*) \le k|\sum w_j C_j$ is feasible if and only if the total processing time of jobs of $\mathcal{J}_N$ scheduled before the last job of $\mathcal{J}_O$ is less than or equal to k.

Proof Similar to the proof of Lemma 1 of Hall and Potts[5].

Lemma 2 For each of the problems

$$1 \mid D_{\max}(\pi^*) \le k \mid \sum w_j C_j, \quad 1 \mid \Delta_{\max}(\pi^*) \le k \mid \sum w_j C_j,$$

there exists an optimal schedule in which the jobs of $\mathcal{J}_O$ are sequenced in SWPT order as in π^*, and there is no idle time between jobs.

Proof Consider an optimal schedule σ^* in which the jobs of $\mathcal{J}_O$ are not sequenced in SWPT order as in π^*. Let J_i be the job with the smallest index that appears later relative to

the other jobs of $\mathcal{J}_O$ in σ^* than in π^*, and let J_j $(j > i)$ be the last job of $\mathcal{J}_O$ that precedes J_i in σ^*. Because π^* is an SWPT schedule, we have $p_i/w_i \leq p_j/w_j$. Since the jobs between J_j and J_i are all new jobs, and they are sequenced by SWPT order, their ratios of processing times and weights are less than p_i/w_i. (Otherwise, interchanging their order, the ratios are less than p_j/w_j.) Consider a new schedule σ' that is obtained from σ^* by interchanging the new jobs between J_i and J_j with J_j successively, at last interchanging J_j and J_i; at each step the objective value of the schedule is not more than the objective value of σ^* and the maximum sequence and time disruption of J_j are less than that of J_i's, respectively. Thus, σ' is feasible and optimal. A finite number of repetitions of this argument shows that there exists an optimal schedule in which the jobs of $\mathcal{J}_O$ are sequenced in SWPT order as in π^*. The same SWPT ordering of the jobs of $\mathcal{J}_O$ in π^*, and an optimal schedule show that there is no idle time in this optimal schedule. Otherwise, removing this idle time, we maintain feasibility and do not increases the total weighted completion time.

Unfortunately, for the rescheduling problems under consideration, there is the possibility that the new jobs of $\mathcal{J}_N$ are not sequenced in SWPT order. For example, suppose $\mathcal{J}_O = \{J_1\}$ and $\mathcal{J}_N = \{J_2, J_3\}$ such that

$$p_1 = 3, \quad p_2 = 1, \quad p_3 = 4, \quad w_1 = 1, \quad w_2 = 1, \quad w_3 = 3, \quad k = 4.$$

Then in an only optimal schedule for the problem $1|\Delta_{\max}(\pi^*) \leq k|\sum w_j C_j$, the three jobs are sequenced in the order (J_3, J_1, J_2), which is not a SWPT order of the new jobs.

But, when p_j and w_j of the jobs are **anticompatible**, the new jobs of $\mathcal{J}_N$ are also sequenced in SWPT order.

Lemma 3 For each of the problems

$$1\,|\,anti-(p_j,w_j), \sum D_j(\pi^*) \leq k\,|\,\sum w_j C_j, \quad 1\,|\,anti-(p_j,w_j), \sum \Delta_j(\pi^*) \leq k\,|\,\sum w_j C_j,$$

there exists an optimal schedule in which the jobs of $\mathcal{J}_O$ are sequenced in SWPT order as in π^*, the jobs of J_N are sequenced in SWPT order, and there is no idle time between jobs.

Proof We first analyze the jobs of $\mathcal{J}_O$. Consider an optimal schedule σ^* in which the jobs of $\mathcal{J}_O$ are not sequenced in SWPT order as in π^*. Let J_i be the job with the smallest index that appears later relative to the other jobs of $\mathcal{J}_O$ in σ^* than in π^*, and let J_j $(j > i)$ be the last job of $\mathcal{J}_O$ that precedes J_i in σ^*. Because π^* is an SWPT schedule, we have $p_i/w_i \leq p_j/w_j$ and $p_i \leq p_j$. Consider a new schedule σ' that is obtained from σ^* by interchanging job J_i and J_j, so that

$$C_i(\sigma') = C_j(\sigma^*) - p_j + p_i, \quad C_j(\sigma') = C_i(\sigma^*),$$

and all jobs between J_j and J_i are completed $p_j - p_i$ units earlier in σ' than in σ^*. Since the jobs between J_j and J_i are all new jobs, and they are sequenced by SWPT order, their ratios of processing time to weight are less than p_i/w_i. (Otherwise, interchanging their order, the ratios are less than p_j/w_j.) Let M denote the set of jobs between J_j and J_i, and $TWC(\sigma)$ denote the total weighted completion time of schedule σ. Then $p_j/w_j \geq p_i/w_i \geq p_r/w_r$, $J_r \in M$, so that

$$p_j/w_j \geq p_i/w_i \geq \sum_{J_r \in M} p_r / \sum_{J_r \in M} w_r.$$

Thus

$$\begin{aligned}
&TWC(\sigma^*) - TWC(\sigma') \\
&= w_jC_j(\sigma^*) + \sum_{J_r\in M} w_rC_r(\sigma^*) + w_iC_i(\sigma^*) - w_jC_j(\sigma') - \sum_{J_r\in M} w_rC_r(\sigma') - w_iC_i(\sigma') \\
&= w_jC_j(\sigma^*) + \sum_{J_r\in M} w_rC_r(\sigma^*) + w_i\Big(C_j(\sigma^*) + \sum_{J_r\in M} p_r + p_i\Big) \\
&\quad -w_j\Big(C_j(\sigma^*) + p_i + \sum_{J_r\in M} p_r\Big) - \sum_{J_r\in M} w_r\big(C_r(\sigma^*) + p_i - p_j\big) - w_i\big(C_j(\sigma^*) - p_j + p_i\big) \\
&= \sum_{J_r\in M} w_r(p_j - p_i) + \sum_{J_r\in M} p_r(w_i - w_j) + w_ip_j - w_jp_i \geq 0.
\end{aligned}$$

This implies that the total weighted completion time does not increase as a result of the interchange. If the position of job J_j in σ' is greater than or equal to its position in π^* (as in the case for job J_i), then $D_j(\pi^*, \sigma') < D_i(\pi^*, \sigma^*)$; otherwise, $D_j(\pi^*, \sigma') < D_j(\pi^*, \sigma^*)$. In either case, because

$$D_i(\pi^*, \sigma') = D_i(\pi^*, \sigma^*) - h \quad \text{and} \quad D_j(\pi^*, \sigma') \leq D_j(\pi^*, \sigma^*) + h,$$

where h is the difference between the positions of jobs J_i and J_j in σ^*, we deduce that

$$D_{\max}(\pi^*, \sigma') \leq D_{\max}(\pi^*, \sigma^*) \quad \text{and} \quad \sum D_j(\pi^*, \sigma') \leq \sum D_i(\pi^*, \sigma^*).$$

Moreover, if $C_j(\sigma') \geq C_j(\pi^*)$, then $\Delta_j(\pi^*, \sigma') < \Delta_i(\pi^*, \sigma^*)$; otherwise $\Delta_j(\pi^*, \sigma') < \Delta_j(\pi^*, \sigma^*)$. In either case, because

$$\Delta_i(\pi^*, \sigma') = \Delta_i(\pi^*, \sigma^*) - h' \quad \text{and} \quad \Delta_j(\pi^*, \sigma') \leq \Delta_j(\pi^*, \sigma^*) + h',$$

where $h' = C_i(\sigma^*) - C_i(\sigma')$, we deduce that

$$\Delta_{\max}(\pi^*, \sigma') \leq \Delta_{\max}(\pi^*, \sigma^*) \quad \text{and} \quad \sum \Delta_j(\pi^*, \sigma') \leq \sum \Delta_i(\pi^*, \sigma^*).$$

Thus, σ' is feasible and optimal. A finite number of repetitions of this argument shows that there exists an optimal schedule in which the jobs of $\mathcal{J}_O$ are sequenced in SWPT order as in π^*. A similar interchange argument establishes that the jobs of J_N can also be sequenced in SWPT order. The same SWPT ordering of the jobs of $\mathcal{J}_O$ in π^*, and an optimal schedule show that there is no idle time in this optimal schedule. Otherwise, removing this idle time, we maintain feasibility and do not increases the total weighted completion time.

The following lemma can be obtained easily.

Lemma 4 For each of the problems

$$1\,|\,anti-(p_j,w_j), D_{\max}(\pi^*) \leq k\,|\,\sum w_jC_j, \quad 1\,|\,anti-(p_j,w_j), \Delta_{\max}(\pi^*) \leq k\,|\,\sum w_jC_j,$$

there exists an optimal schedule in which the jobs of $\mathcal{J}_N$ are sequenced in SWPT order, and there is no idle time between jobs.

Proof Similar to the proof of Lemma 3 by using the condition which p_j and w_j of the new jobs are anticompatible.

The property that the original jobs and the new jobs are scheduled in SWPT order according to their indices, is called the (SWPT, SWPT) property. By Lemma 2, we can find an optimal schedule with the (SWPT, SWPT) property.

3 The algorithm

We consider problem $1|anti-(p_j, w_j), D_{\max}(\pi^*) \leq k|\sum w_j C_j$. Applying the (SWPT, SWPT) property of Lemma 2 and Lemma 1, we have that, up to k jobs of $\mathcal{J}_N$ can be sequenced before the last job of $\mathcal{J}_O$ when p_j and w_j of the new jobs are anticompatible, and these jobs have the smallest ratio of p_j/w_j. Thus, we propose the following algorithm.

Algorithm 1

Input: Given p_j, w_j for $j = 1, \cdots, n$, k and π^*, where $k \leq n_N$.

Indexing: Index the jobs of $\mathcal{J}_N$ in SWPT order.

Coustructing schedule: Schedule jobs $J_1, \cdots, J_{n_O+k}$ in SWPT order in the first $n_O + k$ positions. Schedule jobs $J_{n_O+k+1}, \cdots, J_n$ in SWPT order in the final $n_N - k$ positions.

Theorem 1 Algorithm 1 finds an optimal schedule for problem $1|anti-(p_j, w_j), D_{\max}(\pi^*) \leq k|\sum w_j C_j$ in $O(n + n_N \log n_N)$ time.

Proof From Lemma 1, 2 and 4, the constraint $D_{\max}(\pi^*) \leq k$ allows up to k jobs of $\mathcal{J}_N$ with the smallest ratio of p_j/w_j. Lemma 2 shows that the jobs of this first group are sequenced in SWPT order, Lemma 1 establishes that the remaining $n_N - k$ jobs of $\mathcal{J}_N$ are also sequenced in SWPT order. Note that the Indexing step for the jobs of $\mathcal{J}_N$ requires $O(n_N \log n_N)$ time. The schedule construction step, merging the first k jobs of the SWPT-ordered jobs of $\mathcal{J}_N$ with the jobs of $\mathcal{J}_O$ as sequenced in π^* and then placing the last $n_N - k$ jobs of the SWPT-ordered jobs of $\mathcal{J}_N$ at the end of the schedule, is executed in $O(n)$ time. The result follows.

We next consider problem $1|anti-(p_j, w_j), \Delta_{\max}(\pi^*) \leq k|\sum w_j C_j$. The (SWPT, SWPT) property of Lemma 2 shows that an optimal schedule is found by merging the SWPT-ordered lists of jobs of $\mathcal{J}_O$ and $\mathcal{J}_N$ when p_j and w_j of the new jobs are anticompatible. The following dynamic programming algorithm performs an optimal merging subject to the given constraint on maximum time disruption.

Algorithm 2

Input: Given p_j, w_j for $j = 1, \cdots, n$, k and π^*, where $k \leq P_N$.

Indexing: Index the jobs of $\mathcal{J}_N$ in SWPT order.

Preprocessing: Compute $\sum_{h=1}^{i} p_h$ for $i = 1, \cdots, n_O$ and $\sum_{h=n_O+1}^{n_O+j} p_h$ for $j = 1, \cdots, n_N$.

Value function: $f(i, j, \delta)$ = minimum total weighted completion time of a partial schedule for jobs $J_1, \cdots, J_i$ and $J_{n_O+1}, \cdots, J_{n_O+j}$, where the maximum time disruption is equal to δ.

Boundary condition: $f(0, 0, 0) = 0$.

Optimal solution value: $\min_{0 \leq \delta \leq k} f(n_O, n_N, \delta)$.

Recurrence relation:

$$f(i, j, \delta) = \min \begin{cases} f(i-1, j, \delta - p') + w_i(\sum_{h=1}^{i} p_h + \sum_{h=n_O+1}^{n_O+j} p_h), \\ f(i, j-1, \delta) + w_j(\sum_{h=1}^{i} p_h + \sum_{h=n_O+1}^{n_O+j} p_h). \end{cases}$$

Where p' is the total processing time of jobs of $\mathcal{J}_N$ between jobs J_{i-1} and J_i.

Theorem 2 Algorithm 2 finds an optimal schedule for problem $1|anti-(p_j,w_j),\Delta_{\max}(\pi^*) \leq k|\sum w_jC_j$ in $O(n_On_NP_N+n_N\log n_N)$ time.

Proof From Lemma 1, 2 and 4, We need only enumerate all possible ways of merging the SWPT-order lists of jobs of $\mathcal{J}_O$ and $\mathcal{J}_N$. Algorithm 2 does so by comparing the cost of all possible state transitions, and therefore finds an optimal schedule.

We now consider the time complexity of Algorithm 2. Because $i \leq n_O$, $j \leq n_N$ and $\delta \leq k \leq P_N$, there are $O(n_On_NP_N)$ values of the state variables. Indexing the jobs of $\mathcal{J}_N$ in SWPT order requires $O(n_N\log n_N)$ time. The recurrence relation requires constant time for each set of values of the state variables. Thus, the overall time complexity of Algorithm 2 is $O(n_On_NP_N+n_N\log n_N)$ time.

We next consider problem $1|anti-(p_j,w_j),\sum D_j(\pi^*) \leq k|\sum w_jC_j$. The (SWPT, SWPT) property of Lemma 2 shows that an optimal schedule is found by merging the SWPT-ordered lists of jobs of $\mathcal{J}_O$ and $\mathcal{J}_N$ when p_j and w_j of the jobs are anticompatible. The following dynamic programming algorithm performs an optimal merging subject to the given constraint on total sequence disruption.

Algorithm 3

Input: Given p_j, w_j for $j=1,\cdots,n$, k and π^*, where $k \leq n_On_N$.

Indexing: Index the jobs of $\mathcal{J}_N$ in SWPT order.

Preprocessing: Compute $\sum_{h=1}^{i}p_h$ for $i=1,\cdots,n_O$ and $\sum_{h=n_O+1}^{n_O+j}p_h$ for $j=1,\cdots,n_N$.

Value function: $f(i,j,\delta)$ = minimum total weighted completion time of a partial schedule for jobs $J_1,\cdots,J_i$ and $J_{n_O+1},\cdots,J_{n_O+j}$, where the total sequence disruption is equal to δ.

Boundary condition: $f(0,0,0)=0$.

Optimal solution value: $\min_{0\leq\delta\leq k}f(n_O,n_N,\delta)$.

Recurrence relation:

$$f(i,j,\delta)=\min\begin{cases} f(i-1,j,\delta-j)+w_i(\sum_{h=1}^{i}p_h+\sum_{h=n_O+1}^{n_O+j}p_h), \\ f(i,j-1,\delta)+w_j(\sum_{h=1}^{i}p_h+\sum_{h=n_O+1}^{n_O+j}p_h). \end{cases}$$

Theorem 3 Algorithm 3 finds an optimal schedule for problem $1|anti-(p_j,w_j),\sum D_j(\pi^*) \leq k|\sum w_jC_j$ in $O(n_O^2n_N^2)$ time.

Proof From Lemma 3, We need only enumerate all possible ways of merging the SWPT-order lists of jobs of $\mathcal{J}_O$ and $\mathcal{J}_N$. Algorithm 3 does so by comparing the cost of all possible state transitions, and therefore finds an optimal schedule.

We now consider the time complexity of Algorithm 3. Because $i \leq n_O$, $j \leq n_N$ and $\delta \leq k \leq n_On_N$, there are $O(n_O^2n_N^2)$ values of the state variables. Indexing the jobs of $\mathcal{J}_N$ in SWPT order requires $O(n_N\log n_N)$ time. The recurrence relation requires constant time for each set of values of the state variables. Thus, the overall time complexity of Algorithm 3 is $O(n_O^2n_N^2)$ time.

We next consider problem $1|anti-(p_j,w_j),\sum\Delta_j(\pi^*) \leq k|\sum w_jC_j$. The (SWPT, SWPT) property of Lemma 2 shows that an optimal schedule is found by merging the SWPT-ordered lists of jobs of $\mathcal{J}_O$ and $\mathcal{J}_N$ when p_j and w_j of the jobs are anticompatible. The following

dynamic programming algorithm performs an optimal merging subject to the given constraint on total time disruption.

Algorithm 4

Input: Given p_j, w_j for $j = 1, \cdots, n$, k and π^*, where $k \leq n_O P_N$.

Indexing: Index the jobs of $\mathcal{J}_N$ in SWPT order.

Preprocessing: Compute $\sum_{h=1}^{i} p_h$ for $i = 1, \cdots, n_O$ and $\sum_{h=n_O+1}^{n_O+j} p_h$ for $j = 1, \cdots, n_N$.

Value function: $f(i, j, \delta)$ = minimum total weighted completion time of a partial schedule for jobs $J_1, \cdots, J_i$ and $J_{n_O+1}, \cdots, J_{n_O+j}$, where the total time disruption is equal to δ.

Boundary condition: $f(0, 0, 0) = 0$.

Optimal solution value: $\min_{0 \leq \delta \leq k} f(n_O, n_N, \delta)$.

Recurrence relation:

$$f(i,j,\delta) = \min \begin{cases} f(i-1, j, \delta - \sum_{h=n_O+1}^{n_O+j} p_h) + w_i(\sum_{h=1}^{i} p_h + \sum_{h=n_O+1}^{n_O+j} p_h), \\ f(i, j-1, \delta) + w_j(\sum_{h=1}^{i} p_h + \sum_{h=n_O+1}^{n_O+j} p_h). \end{cases}$$

Theorem 4 Algorithm 4 finds an optimal schedule for problem $1|anti-(p_j, w_j), \sum \Delta_j(\pi^*) \leq k| \sum w_j C_j$ in $O(n_O^2 n_N P_N + n_N \log n_N)$ time.

Proof From Lemma 3, We need only enumerate all possible ways of merging the SWPT-order lists of jobs of $\mathcal{J}_O$ and $\mathcal{J}_N$. Algorithm 4 does so by comparing the cost of all possible state transitions, and therefore finds an optimal schedule.

We now consider the time complexity of Algorithm 4. Because $i \leq n_O$, $j \leq n_N$, and $\delta \leq k \leq n_O P_N$, there are $O(n_O^2 n_N P_N)$ values of the state variables. Indexing the jobs of $\mathcal{J}_N$ in SWPT order requires $O(n_N \log n_N)$ time. The recurrence relation requires constant time for each set of values of the state variables. Thus, the overall time complexity of Algorithm 2 is $O(n_O^2 n_N P_N + n_N \log n_N)$ time.

4 Conclusions

This paper studies the rescheduling problems which the unexpected arrival of new jobs, being taken into account the effect of the disruption on a previously planned optimal schedule. The effect of disruption is measured either by the extent of repositioning within the processing sequence, or by the change in completion time for the original jobs. We consider the rescheduling problem for jobs on a single machine to minimize total weighted completion time under a limit of the disruptions. We show that the considered problem can be solved either in polynomial time or in pseudopolynomial time when p_j and w_j of the jobs are anticompatible.

References:

[1] Wu S D, Storer R H, Chang P C. A rescheduling procedure for manufacturing systems under random disruptions[C]// New Directions for Operations Research in Manufacturing (G. Fandel, T. Gulledge, A. Jone, eds.) Springer, Berlin, Germany, 1992: 292-308

[2] Unal A T, Uzsoy R, Kiran A S. Rescheduling on a single machine with part-type depend setup times and deadlines[J]. Annals of Operations Research, 1997, 70: 93-113

[3] Li W H. A kind of single machine parallel batch scheduling problems with three hierarchical criteria[J]. Chinese Journal of Engineering Mathematics, 2007, (01): 183-186
[4] Mu Y D, Gu C C. Rescheduling with the fixed order[J]. Henan Science, 2007, (01): 8-10
[5] Hall N G, Poots C N. Rescheduling for new orders[J]. Operations Research, 2004, 52: 440-453
[6] Yuan J J, Mu Y D. Rescheduling with release dates to minimize makespan under a limit on the maximum sequence disruption[J]. European Journal of Operation Research, 2007, 182: 936-944
[7] Mu Y D, Yuan J J. Rescheduling to minimize maximum lateness and total tardiness under compatible job systems[J]. OR Transactions, 2007, 1: 39-48
[8] Brucker P. Scheduling Algorithms[M]. Berlin: Springer Verlag, 2001
[9] Graham R L, Lawler E L, Lenstra J K, *et al.* Optimization and approximation in deterministic machine scheduling: a survey[J]. Annals of Discrete Mathematics, 1979, 5: 287-326

反相容工件系统的加权完工时间和的重新排序问题

慕运动[1,2], 谷存昌[1,2], 周 伟[1], 程 瑶[1]

(1- 河南工业大学理学院，郑州 450001; 2- 郑州大学数学系，郑州 450052)

摘 要: 重新排序问题是指在原始工件已经安排好的情形下，新到的工件集与原始工件集一起重新再排序，这是实际工作中常见一类优化问题。本文考虑了单机上当工件加工时间与权重反相容时，在最大错位量约束下的加权完工时间和最小化的重新排序问题。对于提出的四个问题，即在最大序列错位、最大时间错位、总序列错位和总时间错位约束下的加权完工时间和重新排序，基于问题的结构性质，运用动态规划方法分别给出了这些问题的多项式时间或拟多项式时间算法。

关键词: 重新排序；单机；完工时间；错位量；反相容